AMENDMENTS TO THE CLAIMS

Please amend Claims 1, 5, 6, 10, 11, 15, 16, 20, 21, 25, 26, 30, 31, 35, 36, 40, 41, 45, 46, 50, 51, 55, 56 and 60, all as shown below. All pending claims are reproduced below, including those that remain unchanged.

1. (Currently amended) A system run on one or more processors for stopping threads in a safe state in a run-time environment comprising:

a plurality of application threads; and,

a native code interpreter which is configured to stop execution of an executing thread such that the executing thread is stopped, and the native code interpreter interprets machine code to determine if the executing thread is in a safe state, and wherein if the executing thread is not in a safe state, the native code interpreter moves the executing thread forward in steps and at each step determines the state until the executing thread is finally stopped in a safe state.

2. (Original) The system of claim 1 wherein the system includes a virtual machine and wherein said plurality of application threads execute as part of said virtual machine.

3. (Original) The system of claim 1 wherein the system is used for the garbage collection of inactive threads in the run-time environment.

4. (Original) The system of claim 1 wherein the system is used to perform context-switching between the threads in a run-time environment.

5. (Currently amended) The system of claim 1 wherein the native code interpreter is configured to interpret the machine code currently at the executing second thread, and provide [[that]] interpreted machine code information to the system for use in stopping the executing second thread in a safe state.

6. (Currently amended) A system run on one or more processors for moving threads to a safe state in a run-time environment comprising:

a plurality of application threads; and,

a native code interpreter which is used to allow a first or a stopping thread to roll a second or an executing thread forward such that the executing thread is stopped, and the native code interpreter interprets machine code to determine if the executing thread is in a safe state, and wherein if the executing thread is not in a safe state, the native code interpreter moves the executing thread forward in steps and at each step determines the state until the executing thread is finally stopped in a safe state.

7. (Original) The system of claim 6 wherein the system includes a virtual machine and wherein said stopping and executing threads execute as part of said virtual machine.

8. (Original) The system of claim 6 wherein the system is used for the garbage collection of inactive threads in the run-time environment.

9. (Original) The system of claim 6 wherein the system is used to perform context- switching between the threads in a run-time environment.

10. (Currently amended) The system of claim 6 wherein the native code interpreter is configured to interpret the machine code currently at the executing second thread, and provide [[that]] interpreted machine code information to the system for use in stopping the executing second thread in a safe state.

11. (Currently amended) A system run on one or more processors which uses native code interpretation to move threads to a safe state in a run-time environment comprising:

a first and second application threads; and,

a native code interpreter configured to allow the first thread to stop execution of the second thread and roll the second thread forward such that the second thread is stopped, and

the native code interpreter interprets machine code to determine if the second thread is in a safe state, and wherein if the second thread is not in a safe state, the native code interpreter moves the second thread forward in steps and at each step determines the state until the second thread is finally stopped in a safe state.

12. (Original) The system of claim 11 wherein the system includes a virtual machine and wherein said first and second application threads execute as part of said virtual machine.

13. (Original) The system of claim 11 wherein the system is used for the garbage collection of inactive threads in the run-time environment.

14. (Original) The system of claim 11 wherein the system is used to perform context- switching between the threads in a run-time environment.

15. (Currently amended) The system of claim 11 wherein the native code interpreter is configured to interpret the machine code currently at the executing second thread, and provide [[that]] interpreted machine code information to the system for use in stopping the executing second thread in a safe state.

16. (Currently amended) A system run on one or more processors which uses native code interpretation to stop threads in a safe state in a run-time environment, including instructions stored thereon which when executed cause the system to perform the steps of:

allowing a first thread to initially halt execution of a second thread;

using native code interpretation to determine the current state of the second thread; and,

allowing the first thread to roll forward the state of the second thread, wherein if the second thread is not in a safe state, the first thread uses a native code interpreter to move the second thread forward in steps and each step to determine the state until [[such that]] the second thread is finally stopped in a safe state.

17. (Original) The system of claim 16 wherein the system includes a virtual machine and wherein said plurality of first and second threads execute as part of said virtual machine.

18. (Original) The system of claim 16 wherein the system is used for the garbage collection of inactive threads in the run-time environment.

19. (Original) The system of claim 16 wherein the system is used to perform context- switching between the threads in a run-time environment.

20. (Currently amended) The system of claim 16 wherein the native code interpretation is performed by interpreting the machine code currently at the executing second thread, and providing [[that]] interpreted machine code information to the system for use in stopping the executing second thread in a safe state.

21. (Currently amended) A method run on one or more processors for stopping threads in a safe state in a run-time environment, comprising the steps of:

providing a plurality of application threads; and,

providing a native code interpreter which is configured to stop execution of an executing thread such that the executing thread is stopped, and the native code interpreter interprets machine code to determine if the executing thread is in a safe state, and wherein if the executing thread is not in a safe state, the native code interpreter moves the executing thread forward in steps and at each step determines the state until the executing thread is finally stopped in a safe state.

22. (Original) The method of claim 21 wherein the system includes a virtual machine and wherein said plurality of application threads execute as part of said virtual machine.

23. (Original) The method of claim 21 wherein the system is used for the garbage collection of inactive threads in the run-time environment.

24. (Original) The method of claim 21 wherein the system is used to perform context- switching between the threads in a run-time environment.

25. (Currently amended) The method of claim 21 wherein the native code interpreter is configured to interpret the machine code currently at the executing second thread, and provide [[that]] interpreted machine code information to the system for use in stopping the executing second thread in a safe state.

26. (Currently amended) A method run on one or more processors for moving threads to a safe state in a run-time environment, comprising the steps of:

providing a plurality of application threads; and,

providing a native code interpreter which is used to allow a first or a stopping thread to roll a second or an executing thread forward such that the executing thread is stopped, and the native code interpreter interprets machine code to determine if the executing thread is in a safe state, and wherein if the executing thread is not in a safe state, the native code interpreter moves the executing thread forward in steps and at each step determines the state until the executing thread is finally stopped in a safe state.

27. (Original) The method of claim 26 wherein the system includes a virtual machine and wherein said stopping and executing threads execute as part of said virtual machine.

28. (Original) The method of claim 26 wherein the system is used for the garbage collection of inactive threads in the run-time environment.

29. (Original) The method of claim 26 wherein the system is used to perform context- switching between the threads in a run-time environment.

30. (Currently amended) The method of claim 26 wherein the native code interpreter is configured to interpret the machine code currently at the executing second thread, and provide [[that]] interpreted machine code information to the system for use in stopping the executing second thread in a safe state.

31. (Currently amended) A method run on one or more processors which uses native code interpretation to move threads to a safe state in a run-time environment, comprising the steps of:

providing a first and second application threads; and,

providing a native code interpreter configured to allow the first thread to stop execution of the second thread and roll the second thread forward such that the second thread is stopped, and the native code interpreter interprets machine code to determine if the second thread is in a safe state, and wherein if the second thread is not in a safe state, the native code interpreter moves the second thread forward in steps and at each step determines the state until the second thread is finally stopped in a safe state.

32. (Original) The method of claim 31 wherein the system includes a virtual machine and wherein said first and second application threads execute as part of said virtual machine.

33. (Original) The method of claim 31 wherein the system is used for the garbage collection of inactive threads in the run-time environment.

34. (Original) The method of claim 31 wherein the system is used to perform context- switching between the threads in a run-time environment.

35. (Currently amended) The method of claim 31 wherein the native code interpreter is configured to interpret the machine code currently at the executing second thread,

and provide [[that]] interpreted machine code information to the system for use in stopping the executing second thread in a safe state.

36. (Currently amended) A method run on one or more processors which uses native code interpretation to stop threads in a safe state in a run-time environment, comprising the steps of:

allowing a first thread to initially halt execution of a second thread;

using native code interpretation to determine the current state of the second thread; and,

allowing the first thread to roll forward the state of the second thread, wherein if the second thread is not in a safe state, the first thread uses a native code interpreter to move the second thread forward in steps and at each step to determine the state until [[such that]] the second thread is finally stopped in a safe state.

37. (Original) The method of claim 36 wherein the system includes a virtual machine and wherein said plurality of first and second threads execute as part of said virtual machine.

38. (Original) The method of claim 36 wherein the system is used for the garbage collection of inactive threads in the run-time environment.

39. (Original) The method of claim 36 wherein the system is used to perform context- switching between the threads in a run-time environment.

40. (Currently amended) The method of claim 36 wherein the native code interpretation is performed by interpreting the machine code currently at the executing second thread, and providing [[that]] interpreted machine code information to the system for use in stopping the executing second thread in a safe state.

41. (Currently amended) A computer readable medium including instructions stored thereon which when executed by one or more processors on the computer cause the computer to perform the steps of:

providing a plurality of application threads; and,

providing a native code interpreter which is configured to stop execution of an executing thread such that the executing thread is stopped, and the native code interpreter interprets machine code to determine if the executing thread is in a safe state, and wherein if the executing thread is not in a safe state, the native code interpreter moves the executing thread forward in steps and at each step determines the state until the executing thread is finally stopped in a safe state.

42. (Original) The computer readable medium of claim 41 wherein the system includes a virtual machine and wherein said plurality of application threads execute as part of said virtual machine.

43. (Original) The computer readable medium of claim 41 wherein the system is used for the garbage collection of inactive threads in the run-time environment.

44. The computer readable medium of claim 41 wherein the system is used to perform context- switching between the threads in a run-time environment.

45. (Currently amended) The computer readable medium of claim 41 wherein the native code interpreter is configured to interpret the machine code currently at the executing second thread, and provide [[that]] interpreted machine code information to the system for use in stopping the executing second thread in a safe state.

46. (Currently amended) A computer readable medium including instructions stored thereon which when executed by one or more processors on the computer cause the computer to perform the steps of:

providing a plurality of application threads; and,

providing a native code interpreter which is used to allow a first or a stopping thread to roll a second or an executing thread forward such that the executing thread is stopped, and the native code interpreter interprets machine code to determine if the executing thread is in a safe state, and wherein if the executing thread is not in a safe state, the native code interpreter moves the executing thread forward in steps and at each step determines the state until the executing thread is finally stopped in a safe state.

47. (Original) The computer readable medium of claim 46 wherein the system includes a virtual machine and wherein said stopping and executing threads execute as part of said virtual machine.

48. (Original) The computer readable medium of claim 46 wherein the system is used for the garbage collection of inactive threads in the run-time environment.

49. (Original) The computer readable medium of claim 46 wherein the system is used to perform context- switching between the threads in a run-time environment.

50. (Currently amended) The computer readable medium of claim 46 wherein the native code interpreter is configured to interpret the machine code currently at the executing second thread, and provide [[that]] interpreted machine code information to the system for use in stopping the executing second thread in a safe state.

51. (Currently amended) A computer readable medium including instructions stored thereon which when executed by one or more processors on the computer cause the computer to perform the steps of:

providing a first and second application threads; and,

providing a native code interpreter configured to allow the first thread to stop execution of the second thread and roll the second thread forward such that the second thread is stopped, and the native code interpreter interprets machine code to determine if the second thread is in a safe state, and wherein if the second thread is not in a safe state, the native code interpreter

moves the second thread forward in steps and at each step determines the state until the second thread is finally stopped in a safe state.

52. (Original) The computer readable medium of claim 51 wherein the system includes a virtual machine and wherein said first and second application threads execute as part of said virtual machine.

53. (Original) The computer readable medium of claim 51 wherein the system is used for the garbage collection of inactive threads in the run-time environment.

54. (Original) The computer readable medium of claim 51 wherein the system is used to perform context- switching between the threads in a run-time environment.

55. (Currently amended) The computer readable medium of claim 51 wherein the native code interpreter is configured to interpret the machine code currently at the executing second thread, and provide [[that]] interpreted machine code information to the system for use in stopping the executing second thread in a safe state.

56. (Currently amended) A computer readable medium including instructions stored thereon which when executed by one or more processors on the computer cause the computer to perform the steps of:

allowing a first thread to initially halt execution of a second thread;

using native code interpretation to determine the current state of the second thread; and,

allowing the first thread to roll forward the state of the second thread, wherein if the second thread is not in a safe state, the first thread uses a native code interpreter to move the second thread forward in steps and at each step to determine the state until [[such that]] the second thread is finally stopped in a safe state.

57. (Original) The computer readable medium of claim 56 wherein the system includes a virtual machine and wherein said plurality of first and second threads execute as part of said virtual machine.

58. (Original) The computer readable medium of claim 56 wherein the system is used for the garbage collection of inactive threads in the run-time environment.

59. (Original) The computer readable medium of claim 56 wherein the system is used to perform context- switching between the threads in a run-time environment.

60. (Currently amended) The computer readable medium of claim 56 wherein the native code interpretation is performed by interpreting the machine code currently at the executing second thread, and providing [[that]] interpreted machine code information to the system for use in stopping the executing second thread in a safe state.